SPM project A.Y. 2017–18

M. Danelutto

May 24, 2018–Version 1.0

1 SPM final project

Each student should choose one among the following 6 projects (4 assigned, 2 free choice). The choice has to be sent by email to the professor (mandatory Subject: SPM project choice) and after the project approval (by email) you may start work. In a few cases, I can ask to change the choice, for different reasons.

2 "Application" projects

These projects require:

1. to properly design the parallel structure of the application
2. to provide a reference sequential version
3. to provide a performance model of the (expected) parallel performances
4. to provide two parallel implementations, one using C++ only (threads and relative mechanisms) and one using FastFlow
5. to perform (and report) experiments showing actual performances achieved on the Xeon PHI KNC compared with the expected values from the model

2.1 ACO TSP

The Ant Colony Optimization Traveling Salesman Problem requires to calculate a (approximation of the) minimal path connecting all the cities in a graph. The graph is defined as $G = (V, A)$ where $V \in [0, c]$ represent the cities and $A = (s, d, l)$ are connections between cities represented ad a triple with $s \in [0, c]$ being the first city, $d \in [0, c]$ the second city and $l \in [1, maxlen]$ is the length of the path between the two cities. Are are undirectect. $(1, 4, 10) \in A$ represents a connection between $city_1$ and $city_4$ that you may use to go both from $city_1$ to $city_4$ and from $city_4$ to $city_1$. The length of the connection is 10 units of length. The ACO algorithm can be any ACO variant. You may consult the paper in the project annexes and pick up any variant.

Parameters

- Input (minimal): name of the file with the city graph, number of iterations, parallelism degree
- Output (minimal): parallelism degree, "best" city path computed, length of the path, elapsed time

Annexes Copy of the paper, available on the lessons blog and sample input graphs (small, for functional tests, and larger, for performance tests): you can use for this the collections at `http://www.math.uwaterloo.ca/tsp/world/countries.html`.

2.2 Image "watermarking"

We use the "watermark" term in a simplistic version. A collection of images (JPG) in a directory is processed to add to each image a B&W "stamp" represented by a further image having only black and white pixels (no gray scale pixels). The input set of images and the stamp image have the same size, no need for alignment. The `Cimg` library should be used to read/write/process images. Students may consider a variant that substitutes a pixel corresponding to the black pixel of the "stamp" image with the average value of the original color, mapped to gray scale, and the black. On a gray scale 0-255,

this means that if the original pixel was (color) a $[120, 250, 80]$ it can be converted to gray scale 150 (average of the three color values) and therefore it will be represented as a gray scale 202 (integer average with 255).

Parameters

- Input: file directory name, stamp image file name name, parallelism degree
- Output: parallelism degree, number of images processed, elapsed time

Annexes `CImg.h`, the file to include to obtain library functions: download from `http://cimg.eu/`

2.3 Free choice application

Feel free to propose the teacher any application from your own interest domain, *not already used for other exams/thesis in this master degree*. In this case, before starting, write a short text describing the application (more or less of the size of the text in the previous subsections) and add a similar amount of text with a preliminary analysis of the possibilities related to the parallelization. Send the text to professor, by email, and await approval. Professor may ask to discuss the application during question time, in case.

3 "Framework" projects

These project require to implement a parallel pattern, through the following steps:

1. to properly define the parallel structure of the pattern implementation
2. to provide a performance model of the (expected) parallel performances

3. to provide a parallel implementation using C++ only (threads and the relative mechanisms)

4. optionally, to provide an implementation using low level FastFlow building blocks

5. to perform (and report) experiments showing actual performances achieved on the Xeon PHI KNC compared with the expected values from the model

3.1 Divide&Conquer

Provide a class that can be used to create a parallel executor for a divide and conquer pattern. Divide, base case test, base case solution and conquer functions should be given as parameters either in the class constructor or with further accessory methods. A separate method should be implemented that start the evaluation of the divide and conquer on the provided input parameters. A further constructor parameter or method call should be used to fix the parallelism degree to be used in the execution. All types should be parameterized through the template class parameters. Typically we need a `Typein` for the input data and a `Typeout` for the output data. Test programs (at least 2 different tests) should be provided that use the pattern.

Parameters (test programs)

1. Input (minimal): data dimension, parallelism degree

2. Output (minimal): parallelism degree, data size of the input, elapsed time

Annexes None

3.2 Parallel Prefix

Provide a class that can be used to create a parallel executor that, given a `std::vector<T>` input vector $\langle x_1, \ldots, x_n \rangle$ and a `std:function<T(T,T)>` $\oplus$ input function, computes in parallel the vector

$$\langle x_1, x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, \ldots x_1 \oplus x_2 \oplus x_3 \oplus \ldots \oplus x_{n-1} \oplus x_n \rangle$$

A separate method should be implemented that start the evaluation of the parallel prefix on the provided input parameters. A further constructor parameter or method call should be used to fix the parallelism degree to be used in the execution. All types should be parameterized through the template class parameters. Test programs must be provided that use the pattern.

Parameters (test programs)

1. Input (minimal): data dimension, parallelism degree
2. Output (minimal): parallelism degree, data size of the input, elapsed time

Annexes None

3.3 Free choice pattern

Feel free to propose the teacher any different pattern to implement. In this case, before starting, write a short text describing the pattern (more or less of the size of the text in the previous subsections) and add a similar amount of text with a preliminary analysis of the possibilities related to the parallelization. Send the text to professor, by email, and await approval. Professor may ask to discuss the proposed pattern during question time, in case.

4 Report

A report of maximum 10 pages has to be prepared described all the points (general idea and notable implementation details) of the points listed at the beginning of Sec. 2 and 3, that is including (at least) these sections:

- parallel architecture design
- performance modeling
- implementation structure and (notable) implementation details
- experimental validation (with plots)

Any graphs included should be B&W (no colors, use different types of lines (full, dashed, etc.) in case of multiple plots).

5 Exam procedure

For the term whose exam day is reported as "Day X" on the `esami.unipi.it` portal:

- by "day X" send an email message (mandatory Subject: SPM project submission) with your name, enrollment number in the body, and with the PDF report and archive of the source files needed to recompile and run the project in attachment. Professor will read messages the subsequent morning, therefore you can send messages any time before 8am of the Day X+1, indeed.
- professor will take some days to "mark" the projects (depends on how much are submitted to the term and on the other exams in the period, but usually the time will be a few days up to one week) then he will send you an email (to the address you used for the submission) with the time of the oral exam
- the oral exam will consist in a) discussion of the project, with demo through a text only terminal provided by the professor connected to the Xeon PHI machine, and b) some questions relative to the material discussed during the course. The project discussion may require to run the tests, to make small modification to the code or to show in the code where and how certain features are implemented. You may be asked to come to the oral exam also in case the project submission has problems (that is, it is not sufficient to pass the exams). In this case during the oral the professor will only ask you questions about project implementation and tell you what you should do to improve the project.
- if by the project done you are still missing something that can be fixed in some more days (a few, no more that a week), send the project "as is" by the deadline and ask an extension, that will be automatically given, unless particular schedules of the professor activity.